

TP PROGRAMMATION PCDUINO EN C 02 SOCKET

Objectifs : Etre capable de programmer en C un PCDUINO pour envoyer des requêtes POST sur le réseau intranet

ok testé

1. Matériel :.....	1
2. Introduction.....	1
3. Requête POST rappel.....	1
4. Socket : rappel.....	2
5. Fonction guide "connect()".....	2
6. Programmation du socket.....	3
7. Test du fonctionnement.....	3
8. Mesure au sniffer "Wireshark".....	5
9. Problèmes rencontrés.....	5
10. Amélioration de la robustesse du programme.....	6
11. Correction.....	6
11.1. Envoi requête POST du pcdino vers serveur web.....	6

1. MATÉRIEL :

PCDUINO en mode ssh en wifi par vnc ou en direct avec un écran/clavier/souris. (voir Démarrage rapide d'un pcdino)

Geany installé sur le pcdino. (Pour l'installation des paquets geany voir internet)

2. INTRODUCTION

On souhaite envoyer une requête POST vers le serveur de base de données de l'intranet.

3. REQUÊTE POST RAPPEL

Les requêtes POST permettent d'envoyer des données entre un client et un serveur. Les données sont

4. SOCKET : RAPPEL

Les sockets servent à communiquer entre deux hôtes appelés Client / Serveur à l'aide d'une adresse IP et d'un port.

SOCKET = IP + PORT

Ces sockets permettront de gérer des flux entrant et sortant afin d'assurer une communication entre les deux (le client et le serveur), soit de manière fiable à l'aide du protocole TCP/IP, soit non fiable mais plus rapide avec le protocole UDP.

Nous allons utiliser le premier mode, le mode TCP/IP.

5. FONCTION GUIDE "CONNECT0"

Extrait de : PROGRAMMATION RÉSEAU /Arnaud Sangnier /sangnier@liafa.univ-paris-diderot.fr

Création d'une socket

- La création d'une socket se fait grâce à :
 - **#include <sys/socket.h >**
 - **int socket(int domaine, int type, int protocol)**
- Pour nous :
 - **domaine** vaudra **PF_INET** (pour IPv4) ou **PF_INET6** (pour IPv6)
 - **type** vaudra **SOCK_STREAM** (pour les sockets TCP)
 - **protocol** spécifie le protocole de communication (mais pour TCP, on peut mettre 0 et le protocole est choisi de façon automatique)
- L'entier renvoyé sera le descripteur utilisé pour communiquer

Schéma Client-Serveur en C

Pour communiquer

- On va envoyer et recevoir des caractères sur le descripteur de socket
- Pour recevoir on va utiliser
- **ssize_t read(int fildes, void *buf, size_t nbyte);**
 - Remplit le buffer **buf**
 - **nbyte** est la taille maximale de **buf**
 - renvoie le nombre de données reçu (-1 si erreur et 0 si la connexion est fermée)
- Pour envoyer on va utiliser
- **ssize_t write(int fildes, void *buf, size_t nbyte);**
 - Même principe que readl en est la taille en octet de buf
 - Pour le dernier argument, si on est en IPv4 et que adresse est de type **struct sockaddr_in**, on pourra mettre **sizeof(struct sockaddr_in)**
 - Quand on a fini la communication, on peut fermer le descripteur de socket avec la commande
 - **int close(int fildes);**

Exemple

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>

int main() {
    struct sockaddr_in address_sock;
    address_sock.sin_family = AF_INET;
    address_sock.sin_port = htons(4242);
    inet_aton("127.0.0.1", &address_sock.sin_addr);

    int descr=socket(PF_INET, SOCK_STREAM, 0);
    int r=connect(descr, (struct sockaddr *)&address_sock,
        sizeof(struct sockaddr_in));

    if(r!=-1){
        char buff[100];
        int size_rec=read(descr, buff, 99*sizeof(char));
        buff[size_rec]='\0';
        printf("Caracteres recus : %d\n", size_rec);
        printf("Message : %s\n", buff);
        char *mess="SALUT!\n";
        write(descr, mess, strlen(mess));
    }
    return 0;
}
```

Autre exemple de Client TCP en mode connecté (*extrait : Programmation réseaux v1.0*)

Le mode connecté (client TCP)

- Créer le point de communication :
socket() int socketDialogue = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
- Déterminer le point de rencontre distant (le serveur) :
remplir une structure sockaddr_in
struct sockaddr_in pointDeRencontreDistant;
socklen_t longueurAdresse;
longueurAdresse = sizeof(pointDeRencontreDistant);
memset(&pointDeRencontreDistant, 0x00, longueurAdresse);
pointDeRencontreDistant.sin_family = PF_INET; pointDeRencontreDistant.sin_port =
htons(IPPORT_USERRESERVED); inet_aton("192.168.52.83",
&pointDeRencontreDistant.sin_addr);
- Se connecter au point de rencontre distant (le serveur) : **connect()**
connect(socketDialogue, (struct sockaddr *)&pointDeRencontreDistant,
longueurAdresse);
- Echanger des données avec le point de rencontre distant (le serveur) :
send()/recv() ou read()/write()
en (définissant et) utilisant un protocole et une procédure d'échange communs
- Fermer la connexion et le point de communication : **close() ou shutdown()**
close(socketDialogue);

A partir de ces exemples nous créons notre programme.

6. PROGRAMMATION DU SOCKET

```

struct sockaddr_in adress_sock;//adresse socket avec type utilisé par connect()
adress_sock.sin_family = AF_INET;//domaine IPV4 = AF_INET
adress_sock.sin_port = htons(80);//port 80
inet_aton("192.168.1.11",&adress_sock.sin_addr);// manipulation d'adresse : IP du serveur en local
int monsocket = socket(PF_INET,SOCK_STREAM,0);
int r=connect(monsocket,(struct sockaddr *)&adress_sock,sizeof(struct sockaddr_in));

if (r!=1)//si connexion ok
{
    //envoi de la requete POST
    write (monsocket,messComplet,strlen(messComplet));
}
}

```

7. TEST DU FONCTIONNEMENT

Afin de tester le bon envoi du socket on utilise "wireshark".

Les conditions de test sont :

- Programme en C sur pcdduino (192.168.1.39) avec geany (IDE C)
- Serveur Wamp sur PC (192.168.1.11:80).
- Base de donnée : projet_db_mesures/dbvisu.php visualise les données de la base db_mesures/table_mesures

The screenshot displays the phpMyAdmin web interface. The main panel shows the 'Structure' tab for the 'table_mesures' table. The table structure is as follows:

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
1	id	int(11)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer plus
2	jourheure	timestamp			Non	Aucune		Modifier Supprimer plus
3	mesure1	int(11)			Non	Aucune		Modifier Supprimer plus
4	mesure2	int(11)			Non	Aucune		Modifier Supprimer plus
5	commentaire	int(11)			Non	Aucune		Modifier Supprimer plus

Below the table structure, there are options for 'Index' and 'Ajouter' (Add) with a dropdown menu set to 'id'. The bottom section of the interface shows 'Information' about the table, including 'Espace utilisé' (Space used) and 'Statistiques' (Statistics).

Espace utilisé		Statistiques	
Données	16 Kio	Format	Compact
Index	0 o	Interclassement	latin1_swedish_ci
Total	16 Kio	Prochain index automatique	24 746
		Création	Dim 03 Avril 2016 à 20:45

8. MESURE AU SNIFFER "WIRESHARK"

The image shows a Wireshark network traffic capture. The top pane displays a list of captured packets, filtered by 'http'. The selected packet (No. 4350) is an HTTP POST request to '/projet_db_mesures/dbvisu.php'. The middle pane shows the details of this packet, including the Hypertext Transfer Protocol section. The bottom pane shows the raw packet data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
3646	45.692760	192.168.1.6	192.168.1.11	HTTP	221	POST /projet_db_mesures/dbvisu.php HTTP/1.1
3674	51.800121	fe80::2958:8d1b:79f1::ff02::c		SSDP	208	M-SEARCH * HTTP/1.1
3710	54.800061	fe80::2958:8d1b:79f1::ff02::c		SSDP	208	M-SEARCH * HTTP/1.1
3723	57.800721	fe80::2958:8d1b:79f1::ff02::c		SSDP	208	M-SEARCH * HTTP/1.1
4259	112.325981	192.168.1.6	192.168.1.11	HTTP	221	POST /projet_db_mesures/dbvisu.php HTTP/1.1
4341	124.267413	192.168.1.254	239.255.255.250	SSDP	314	NOTIFY * HTTP/1.1
4345	124.418498	192.168.1.254	239.255.255.250	SSDP	394	NOTIFY * HTTP/1.1
4346	124.518712	192.168.1.254	239.255.255.250	SSDP	394	NOTIFY * HTTP/1.1
4347	124.619040	192.168.1.254	239.255.255.250	SSDP	394	NOTIFY * HTTP/1.1
4348	124.721372	192.168.1.254	239.255.255.250	SSDP	378	NOTIFY * HTTP/1.1
4349	124.742085	192.168.1.254	239.255.255.250	SSDP	323	NOTIFY * HTTP/1.1
4350	124.771699	192.168.1.254	239.255.255.250	SSDP	380	NOTIFY * HTTP/1.1

Details of packet 4350:

- Checksum: 0x6f36 [validation disabled]
- Urgent pointer: 0
- Options: (12 bytes), No-operation (NOP), No-operation (NOP), Timestamps
 - No-operation (NOP)
 - No-operation (NOP)
 - Timestamps: TSval 956773, TSecr 22644871
 - Kind: Time stamp Option (8)
 - Length: 10
 - Timestamp value: 956773
 - Timestamp echo reply: 22644871
- [SEQ/ACK analysis]
 - [This is an ACK to the segment in frame: 3645]
 - [The RTT to ACK the segment was: 0.007407000 seconds]
 - [Bytes in flight: 155]
- Hypertext Transfer Protocol
 - POST /projet_db_mesures/dbvisu.php HTTP/1.1\r\n
 - [Expert Info (Chat/Sequence): POST /projet_db_mesures/dbvisu.php HTTP/1.1\r\n]
 - POST /projet_db_mesures/dbvisu.php HTTP/1.1\r\n

Raw packet data (hex/ascii):

```

0000 28 18 78 d2 fb 19 cc d2 9b 2f ee ae 08 00 45 00 (.x.... ./....E.
0010 00 cf 76 6e 40 00 40 06 40 59 c0 a8 01 06 c0 a8 ..vn@.@Y.....
0020 01 0b 94 85 00 50 b5 cb f7 99 b1 3e 88 52 80 18 ....P...>.R..
0030 00 73 6f 36 00 00 01 01 08 0a 00 0e 99 65 01 59 ..so6... ..e.Y
0040 88 87 50 4f 53 54 20 2f 70 72 6f 6a 65 74 5f 64 ..POST / projet_d
0050 62 5f 6d 65 73 75 72 65 73 2f 64 62 76 69 73 75 b_mesure s/dbvisu
0060 2e 70 68 70 20 48 54 54 50 2f 31 2e 31 0d 0a 48 .php HTT P/1.1..H
0070 6f 73 74 3a 31 39 32 2e 31 36 38 2e 31 2e 31 31 ost:192. 168.1.11
0080 0d 0a 43 6f 6e 74 65 6e 74 20 54 79 70 65 3a 20 ..Conten t Type:
0090 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 2d 77 77 applicat ion/x-ww
00a0 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e 63 6f 64 65 w-form-u rlenode
00b0 64 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 68 d..Conte nt-Lengh
00c0 74 3a 32 31 0d 0a 0d 0a 6d 65 73 75 72 65 31 3d t:21.... mesure1=
00d0 35 36 26 6d 65 73 75 72 65 32 3d 35 38 56&mesur e2=58
  
```

Le POST apparaît dans l'enregistrement du sniffer.

Cette méthode permet de détecter d'éventuelles erreurs de connexion et de corriger le programme.

9. PROBLÈMES RENCONTRÉS

Problème de syntaxe de la requête :

1. erreur : Content-length : au lieu de Content-length !!!! (2 jours pour vérifier)

10. AMÉLIORATION DE LA ROBUSTESSE DU PROGRAMME

Prévoir le "else" si la connexion ne se fait pas

Prévoir une acquisition des données sur deux canaux analogiques (ou un analogique et un numérique)

Prévoir d'écrire mes mesures dans un fichier

Prévoir d'écrire un fichier log

Prévoir de lancer le programme au démarrage

11. CORRECTION

11.1. Envoi requête POST du pcdduino vers serveur web

Programme écrit avec Geany sous lubuntu sur un pcdduino

Fichier projet prog.c.geany

```
[indentation]
indent_width=4
indent_type=1
indent_hard_tab_width=8
detect_indent=false
detect_indent_width=false
indent_mode=2

[project]
name=prog.c
base_path=/home/ubuntu/projects/prog.c/
description=
file_patterns=

[long line marker]
long_line_behaviour=1
long_line_column=72

[files]
current_page=1
FILE_NAME_0=1140;C;0;16;1;1;0;/home/ubuntu/projects/prog.c/essai00/main.c;0;4
FILE_NAME_1=240;C;0;16;1;1;0;/home/ubuntu/projects/prog.c/essai00/envoiPOST.c;0;4

[VTE]
last_dir=/home/ubuntu
```

EnvoiPOST.c

```
/*envoi requete POST à un serveur en 192.168.1.11 by SB.
//attention pour utiliser modifier le messPOST en fonction de votre base de donnée
et de votre fichier php.
Modifier aussi l'IP du serveur( dans messPOST et aussi dans inet_aton
*/
```

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#include <netinet/in.h> //contient les macro de conversion de données entre les différentes machines (les données
ne sont pas représentées identiquement entre linux et windows...)
#include <arpa/inet.h>
#include <sys/socket.h>
```

```
#include <string.h>
#include <unistd.h>

#include <time.h>

#include "envoiPOST.h"

//partie fixe de la requete à adapter à vos besoins
const char *messPOST="POST /projet_db_mesures/dbvisu.php HTTP/1.1\r\n"
"Host: 192.168.1.11\r\n"
"Content-Type: application/x-www-form-urlencoded\r\n"
"Content-Length: ";

//prototypes de fonction
void attente(float temps);
int acquerirMesuresAnalogiques(int *,int *);

/*****/
void attente (float tps){
    //attente en seconde

    clock_t arrivee = clock()+(tps*CLOCKS_PER_SEC);
    while(clock()<arrivee);

} //attente()
/*****/
int acquerirMesuresAnalogiques(int *lmesure1, int *lmesure2){
    //acquiere les mesures analogiques sur la carte pcdino

    return 0;
} //acquerirMesuresAnalogiques
/*****/
int main_envoiPOST(int mesure1, int mesure2){

    printf("Entrez dans envoiPOST()\r\n");

    struct sockaddr_in adress_sock;//adresse socket avec type utilisé par connect()
    adress_sock.sin_family = AF_INET;//domaine IPV4 = AF_INET
    adress_sock.sin_port = htons(80);//port 80
    inet_aton("192.168.1.11",&adress_sock.sin_addr);// manipulation d'adresse : IP du serveur en local

    //creation de la partie variable de la requete contient les données
    char messData[50]="";
    int messLongData;
        //les données sont converties en char
    char cmesure1[4]="",cmesure2[4]="";
    sprintf(cmesure1,"%d",mesure1);sprintf(cmesure2,"%d",mesure2);
        //concatenation des mesures
    strcat(messData,"\r\n\r\nmesure1=");
    strcat(messData,cmesure1);
    strcat(messData,"&mesure2=");
    strcat(messData,cmesure2);
        //longueur des données en char
    messLongData = strlen(messData)-4; // -4 car on enlève les \r\n\r\n
    char cmessLongData[3]="";
    sprintf(cmessLongData,"%d",messLongData);

    //creation du message complet
```

```
char messComplet[1024]="";
strcat(messComplet,messPOST);
strcat(messComplet,cmessLongData);
strcat(messComplet,messData);
printf("messComplet : %s\r\n",messComplet);
//creation de la requete
int monsocket = socket(PF_INET,SOCK_STREAM,0);
int r=connect(monsocket,(struct sockaddr *)&adress_sock,sizeof(struct sockaddr_in));

if (r!=1)//si connexion ok
{
    //envoi de la requete POST
    write (monsocket,messComplet,strlen(messComplet));
}

return 0;
} //main_envoiPOST
```

Programme principal :

```
/*
 * main.c
 *
 */

#include <stdio.h>
#include "analogin_sb.c"
#include "gpio_sb.c"
#include "envoiPOST.c"

int main(int argc, char **argv)
{
    printf("Envoi POST start ...\r\n");
    while (1){

        printf("essai envoiPOST 2221\r\n");

        main_envoiPOST(33,133); //envoi des valeurs mesure1=33 et mesure2=133 en POST

        //allume LED attente 400ms
        //setPin(pinData[9], LOW); attente(0.4);
        //eteint LED
        //setPin(pinData[9],HIGH); attente(0.4);

        attente(10);

    } //while
    return 0;
}
```

Fichier envoiPOST().h

```
/******
```

```
* envoiPOST.h
```

```
modifie par sb
```

```
*
```

```
Header file for pcDuino GPIO example code.
```

```
26 Mar 2013 - Mike Hord, SparkFun Electronics
```

```
This code is beerware- if you find it useful, please buy me (or, for that  
matter, any other SparkFun employee you met) a pint next time you meet us at  
the local.
```

```
*****/
```

```
#ifndef __gpio_sb_h__
```

```
#define __gpio_sb_h__
```

```
// These are the paths and filenames of the files for GPIO access.
```

```
#define GPIO_MODE_PATH "/sys/devices/virtual/misc/gpio/mode/"
```

```
#define GPIO_PIN_PATH "/sys/devices/virtual/misc/gpio/pin/"
```

```
#define GPIO_FILENAME "gpio"
```

```
void writeFile(int fileID, int value);
```

```
void setPinMode(int pinID, int mode);
```

```
void setPin(int pinID, int state);
```

```
int main_envoiPOST();
```

```
// Make some aliases to make the code more readable.
```

```
#define HIGH '1'
```

```
#define LOW '0'
```

```
#define INPUT '0'
```

```
#define OUTPUT '1'
```

```
#define INPUT_PU '8'
```

```
#endif
```